KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

KOMPIUTERIŲ KATEDRA

Audrys Kažukauskas

Tekstinių dokumentų išsaugojimo ir išrinkimo metodų dokumentų valdymo sistemoje tyrimas

Magistro darbas

Darbo vadovas

doc. dr. E. Kazanavičius

Kaunas, 2004

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. E. Kazanavičius
2004 05

Tekstinių dokumentų išsaugojimo ir išrinkimo metodų dokumentų valdymo sistemoje tyrimas

Informatikos mokslo magistro baigiamasis darbas

Kalbos konsultantė
Lietuvių kalbos katedros lektorė
dr. J. Mikelionienė
2004 05

Vadovas
doc. dr. E. Kazanavičius
2004 05

Recenzentas
doc. dr. S. Maciulevičius
2004 05

Atliko
IFM-8/1 gr. stud.
Audrys Kažukauskas
2004 05

Kaunas, 2004

Turinys

Įvadas

Dokumentų valdymas yra automatizuotas elektroninių dokumentų valdymas per visą jų gyvavimo ciklą – nuo dokumentų sukūrimo, išsaugojimo, išrinkimo iki sunaikinimo ar archyvavimo [1]. Dokumentų valdymo sistema yra automatizuota elektroninė sistema, kuri realizuoja dokumentų valdymą. Dokumentų valdymą dažniausiai sudaro tokie procesai: įvedimas, apdorojimas, integravimas, išsaugojimas, išrinkimas ir paskirstymas.

Šiame darbe tiriami tekstinių dokumentų išsaugojimo ir išrinkimo procesai.

Pagrindinis darbo tikslas – parinkti metodą tekstiniams dokumentams išsaugoti ir išrinkti, tinkamą dokumentų valdymo sistemai.

Darbo tikslui įgyvendinti sprendžiami tokie uždaviniai:

1. parinkti dokumentų atvaizdavimo formatą,
2. parinkti dokumentų saugyklą,
3. parinkti arba sukurti dokumentų išsaugojimo ir išrinkimo metodą.

Šio darbo 1 skyriuje analizuojami dokumentų valdymo sistemos procesai, tekstinių dokumentų formatai bei jų ypatumai, alternatyvios dokumentų saugyklos bei egzistuojantys metodai tekstiniams dokumentams išsaugoti ir išrinkti.

Siūlomas dokumentų išsaugojimo ir išrinkimo metodo teorinis modelis - modifikuotasis briaunų metodas yra detaliai aprašytas 2 skyriuje. Pateikiamas konkretaus dokumento išsaugojimo ir išrinkimo pagal šį metodą pavyzdys.

Eksperimentinis modelis modifikuotojo briaunų metodo efektyvumui patikrinti ir eksperimento metu gauti duomenys yra pateikti 3 skyriuje. Darbo pabaigoje pateikiamos išvados.

1. Dokumentų išsaugojimo ir išrinkimo procesų analizė

Šiame skyriuje analizuojami dokumentų valdymo sistemos procesai, tekstinių dokumentų formatai, alternatyvios dokumentų saugyklos bei egzistuojantys metodai tekstiniams dokumentams išsaugoti ir išrinkti.

1.1 Dokumentų valdymo sistemos apibrėžimas

Dokumentų valdymo sistema (DVS) – tai automatizuota elektroninė sistema, kuri:

- ✓ suteikia galimybę pagal nustatytą gyvavimo ciklą kurti, kaupti ir sisteminti organizacijai svarbius duomenis;
- ✓ turi įrankius, kuriais iš sukauptų duomenų galima išgauti aktualią informaciją, padedančią organizacijos darbuotojams greitai reaguoti į susidariusią situaciją bei priimti tinkamą sprendimą;
- ✓ užtikrina kolektyvinį priėjimą prie informacijos. Daug naudotojų, net iš skirtingų pasaulio vietų, gali vienu metu operuoti ta pačia informacija;
- ✓ patikimai užtikrina prieigą prie informacijos pagal suteiktas teises. Naudotojas gali prieiti prie informacijos ir operuoti ja tik tiek, kiek leidžia jam suteiktos teisės.

Sekančiame skyriuje aptariami dokumentų valdymą sudarantys procesai.

1.2 Dokumentų valdymo procesai

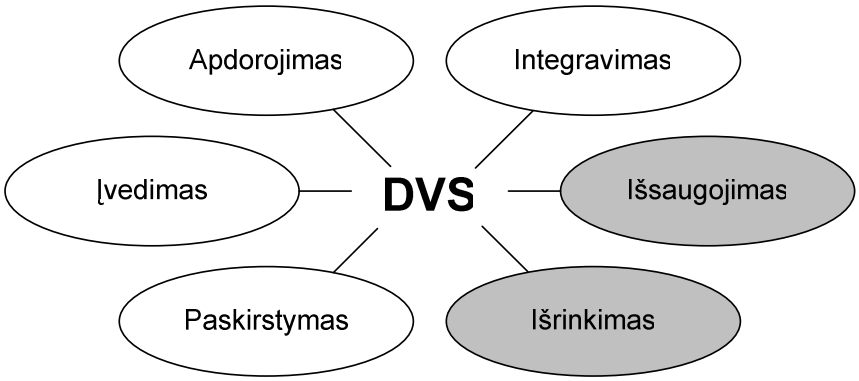
Išskiriami tokie dokumentų valdymo procesai [8]:

- ✓ įvedimas,
- ✓ apdorojimas,
- ✓ integravimas,
- ✓ išsaugojimas,
- ✓ išrinkimas,
- ✓ paskirstymas.

Dokumentai į DVS įkeliami elektroniniu formatu. Prie elektroninio formato priskiriamas ir atvejis, kai dokumentas yra sukuriamas užpildant elektroninę formą. Jei dokumentas yra popierinis, pirmiausia jis yra konvertuojamas į elektroninę formą. Šiuolaikinė skenavimo programinė įranga yra pakankamai ištobulėjusi. Ji leidžia vienu klavišo

spustelėjimu nuskenuoti ir tuo pačiu automatiškai sureguliuoti skenavimo kokybę. Skenuotas dokumentas yra konvertuojamas iš taškinio į tekstinį formatą.

Į DVS įvesti dokumentai yra apdorojami pagal organizacijos dokumentų tvarkymo taisykles, kurios paprastai atitinka organizacijos vykdomą kokybės politiką. Dauguma organizacijų turi savitas dokumentų kūrimo, patvirtinimo (dar vadinama *vizavimu*), audito taisykles, savitą dokumento gyvavimo ciklą.



1 pav. Dokumentų valdymo procesai

Įvesti ir apdoroti dokumentai yra integruojami į sistemą. Integravimas yra labai svarbus etapas, nes jis suteikia galimybę vykdyti dokumentų paiešką ir juos peržiūrėti iš organizacijos LOB (angl. *Line-of-Business*) sistemos pusės. Šiuolaikinės DVS gali būti visiškai integruotos į LOB sistemas [8].

Dokumentai yra išsaugomi elektroniniu formatu dokumentų saugykloje: duomenų bazių valdymo sistemoje arba failinėje sistemoje. Įmanomas variantas dokumentus išsaugoti panaudojus kompleksinį sprendimą: patį dokumento failą laikyti failinėje sistemoje, o dokumentą aprašančius metaduomenis – duomenų bazių valdymo sistemoje. Išsaugojimo terpės pasirinkimas priklauso nuo konkrečių sistemos reikalavimų. Tai gali būti standieji diskai serveryje, kompaktiniai diskai.

Dokumentų išrinkimas ir paskirstymas yra momentinis ir globalus. Užtikrinus reikiamą saugumo lygį, dokumentai yra prieinami iš bet kurios pasaulio vietos (pvz., per interneto naršyklę).

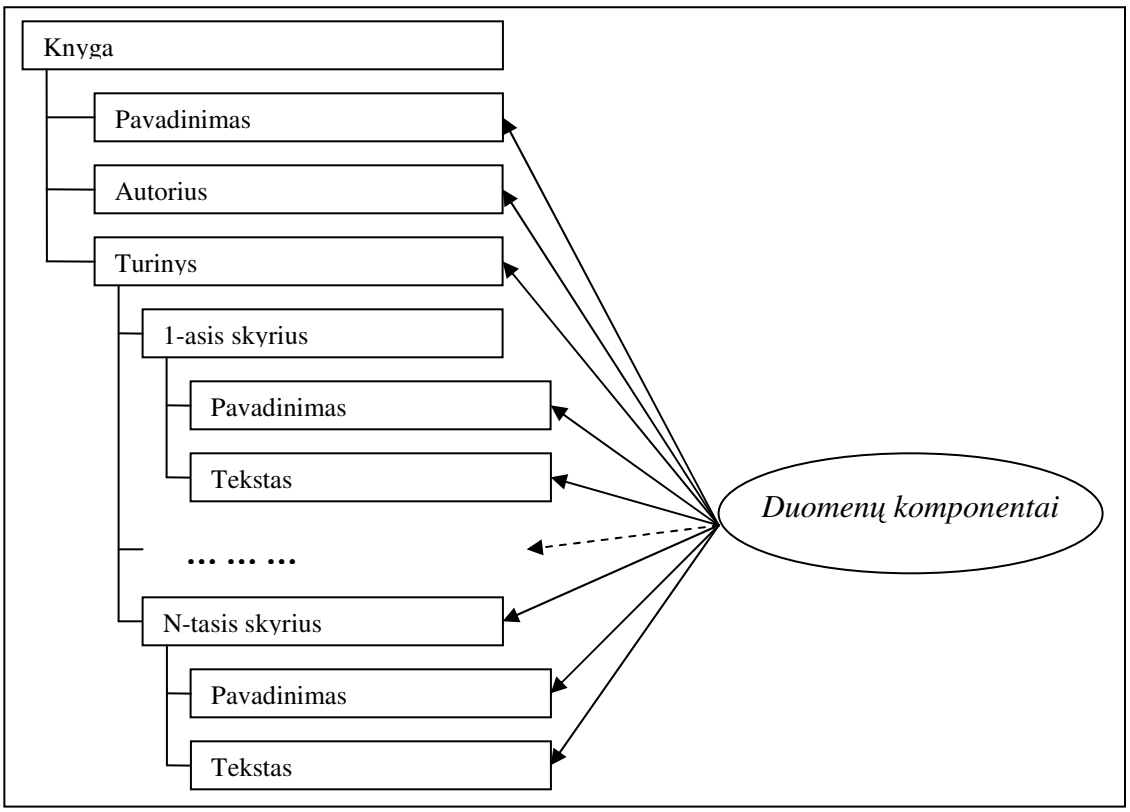
Šiame darbe nagrinėjami šie dokumentų valdymo procesai: **išsaugojimas ir išrinkimas** (1 pav.).

1.3 Dokumentų struktūra

DVS operuoja dokumentais. Šiame darbe tiriami tik tekstiniai dokumentai (toliau jie bus vadinami tiesiog dokumentais).

Dokumentas yra tarpusavyje glaudžiai susijusių duomenų rinkinys, kuris yra siejamas bendro konteksto bei turintis aiškią pradžią ir pabaigą. Paprastai dokumentą sudaro daug atskirų duomenų rinkinių. Tokie atskiri duomenų rinkiniai toliau bus vadinami duomenų komponentais.

Duomenų komponentai dokumente yra išsidėstę hierarchiškai ir sudaro medžio pavidalo struktūrą. Kaip hierarchinio dokumento pavyzdį galima pateikti apibendrintą knygos modelį (2 pav.). Kiekviena knyga turi tokius duomenų komponentus kaip pavadinimas, autorius ir kt.



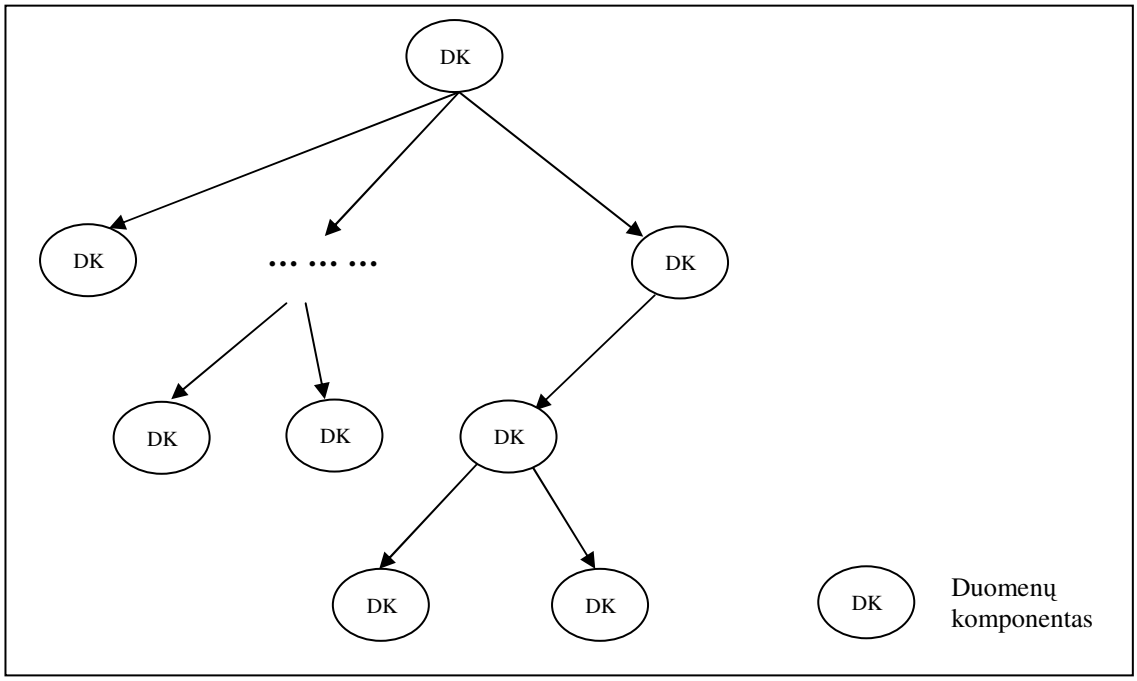
2 pav. Apibendrinta knygos hierarchinė struktūra

Atskiras duomenų komponentas gali būti siejamas metaduomenų komponentu. Pastarasis aprašo duomenų komponento atributus, pvz., formatavimo stilių (šrifto dydį, šrifto tipą ir kt.).

Hierarchines duomenų struktūras patogu aprašyti medžio pavidalo grafais (toliau nagrinėjami tik medžio pavidalo grafai, kurie vadinami tiesiog grafais) [7,9].

Grafo *G* elementai yra dvi aibės: viršūnių aibė *V* ir briaunų aibė *E*. Tarp *V* ir *E* yra apibrėžtas incidentumo santykis: kiekviena briauna (elementas iš *E*) atitinka dvi ir tik dvi viršūnes (elementų iš *V* porą, elementų tvarka poroje fiksuota)[10].

Grafe *G* duomenų komponentai yra atvaizduojami viršūnėmis v_i, o jų tarpusavio ryšiai – briaunomis e_j (3 pav.). Viena viršūnė medyje išskiriama kaip šakninė viršūnė. Ji vienintelė visame grafe *G* neturi tėvo.



3 pav. Hierarchinės duomenų struktūros atvaizdavimas grafu

Dokumento duomenų komponentų grafas *G* pasižymi tokiomis savybėmis:

- ✓ *neturi izoliuotų viršūnių*:

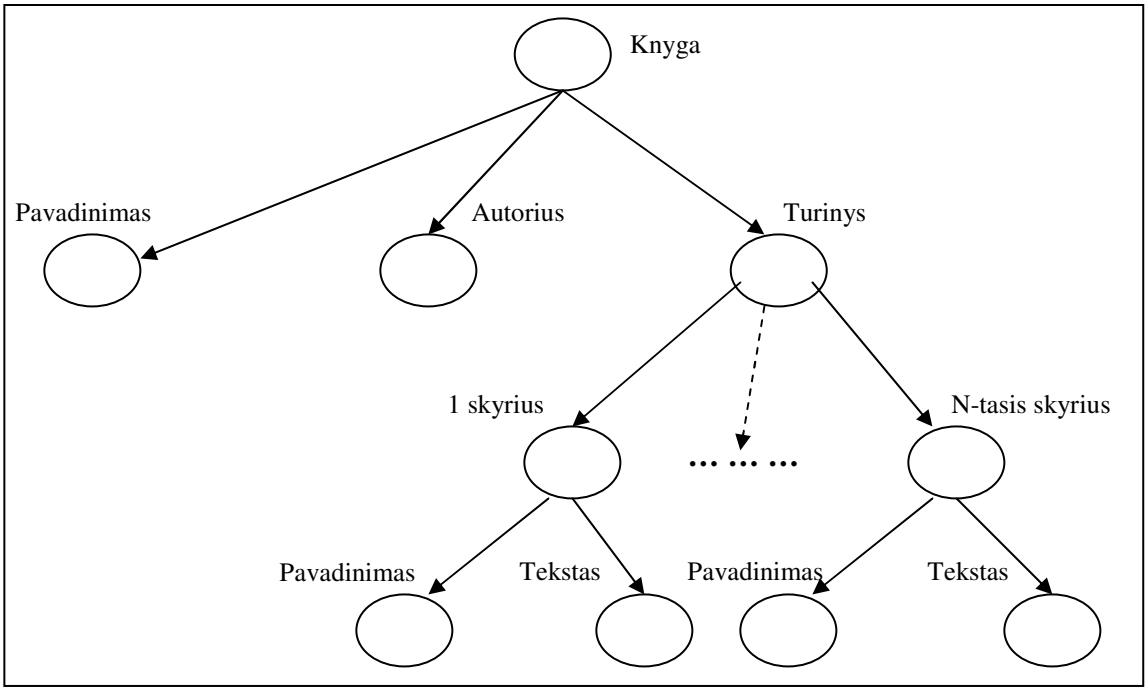
  $deg\ v_i \neq 0,\ \forall v_i \in G,\ kai\ i=1..N,$

  kur $deg\ v_i$ – *i*-tosios viršūnės laipsnis,

  N – grafo *G* viršūnių skaičius;
- ✓ *yra jungus*, t. y. bet kurios dvi jo viršūnės sujungtos keliu;
- ✓ *yra orientuotas*, tarp dviejų gretimų viršūnių yra apibrėžta priklausomybės kryptis:

pr, *pb*: $E \rightarrow V$, kur *pr*(*e*) yra briaunos *e* pradžia, o *pb*(*e*) yra briaunos *e* pabaiga [11].

Pavyzdyje (2 pav.) pateiktos knygos duomenų komponentų struktūros grafas yra pavaizduotas 4 pav. Šio grafo gylis *t=4* [12]. Pirmajame lygyje yra tik viena šakninė viršūnė *Knyga*. Antrojo lygio viršūnės *Pavadinimas*, *Autorius*, *Turinys* yra šakninės viršūnės *Knyga* vaikai. Antrojo lygio viršūnės *Pavadinimas* ir *Autorius* yra lapai (Lapas – tai viršūnė, kurių vaikų skaičius yra lygus nuliui). Antrojo lygio viršūnė *Turinys* yra vidinė viršūnė (vidinėmis vadinamos visos viršūnės, išskyrus šakninę viršūnę ir lapus) ir turi vaikus - trečiojo lygio viršūnes *1-asis skyrius, ..., N-tasis skyrius*. Knygos grafo hierarchijos apačioje yra viršūnės *Pavadinimas* ir *Tekstas*.



4 pav. Knygos (2 pav.) duomenų struktūros grafas

Čia pateikta supaprastinta knygos struktūra. Realių dokumentų duomenų komponentai yra smulkesni, todėl tokių dokumentų grafų gylis *t* yra didesnis ir jie pasižymi didesniu viršūnių skaičiumi.

1.4 Dokumentų tipai

Pagal duomenų struktūros ypatumus egzistuoja du dokumentai tipai:

- ✓ į duomenis orientuoti (angl. *data-centric*),
- ✓ į dokumentus orientuoti (angl. *document-centric*).

Į duomenis orientuoti dokumentai – tai dokumentai, kurių duomenų komponentai išsidėstę pagal griežtai apibrėžtas taisykles. Paprastai jie yra kuriami įvedant duomenis į šabloną (elektroninę formą) arba formuojami iš duomenų, kurie saugomi duomenų bazėse. Pirmu atveju, kaip pavyzdys tinka mokesčių mokėjimas ar prekės užsakymas internetu. Antruoju atveju, kaip pavyzdį galima pateikti įmonės pardavimų ataskaitas su statistine informacija apie per mėnesį parduotus produktus.

Į dokumentus orientuoti dokumentai yra tik dalinai struktūrizuoti arba iš viso nestruktūrizuoti. Tokie dokumentai gali neturi duomenų komponentų išdėstymo taisyklių arba jos gali dažnai keistis. Tai knygos, laiškai ir el. pašto žinutės, skelbimai ir beveik kiekvienas kitas ranka rašytas dokumentas. Duomenų komponentų eiliškumo tvarka į šio tipo dokumentuose yra svarbi.

1.5 Dokumentų formatai

Šiame skyriuje analizuojami egzistuojantys dokumentų formatų standartai.

1.5.1 RTF formatas

RTF (angl. *Rich Text Format*) yra specifikacija, kuri apibrėžia tekstinį failo formatą, skirtą apsikeisti tekstu ir grafika tarp įvairių išvesties įrenginių, programų ir operacinių sistemų [18]. Nors RTF specifikacija sukurta Microsoft korporacijos, apsikeisti dokumentais galima ne tik tarp įvairių MS-DOS®, Microsoft® Windows® (pvz., Windows 98 operacinėje sistemoje su Microsoft Word programa sukurtas ir išsaugotas RTF (su plėtiniu *.RTF) dokumentas gali būti perskaitytas Windows 3.1 operacinėje sistemoje su WordPerfect 6.0 programa), OS/2, Macintosh®, ir Power Macintosh®, bet ir UNIX, Apple programų.

RTF dokumentas yra sudarytas iš neformatuoto teksto, valdančiųjų žodžių, valdančiųjų simbolių ir grupių [18]. Dokumentų transportavimui palengvinti, standartinį RTF dokumentą sudaro tik 7 bitų ASCII simboliai.

Valdantysis žodis yra specialaus formato komanda, žyminti spausdintuvo valdymo kodus ir informaciją, kurią programos naudoja dokumentams apdoroti. Valdančiojo žodžio ilgis negali viršyti 32 simbolių.

Valdantieji simboliai yra sudaryti iš įžambaus brūkšnio simbolio ir vieno nealfabetinio simbolio. Pavyzdžiui, \~ reiškia tarpą.

Grupė yra sudaryta iš teksto ir valdančiųjų žodžių arba simbolių. Grupė prasideda skliaustu ({) ir baigiasi skliaustu (}). Kiekviena grupė apibrėžia joje esantį tekstą ir jo atributus. RTF dokumentas taip pat gali turėti šriftų, stilių, spalvų, komentarų ir kitas grupes, taip pat dokumento, skirsnio, paragrafo ir atskiro simbolio formatavimą žyminčius atributus. Šrifto, stiliaus, dokumento formatavimo ir kiti atributai turi būti aprašyti prieš dokumento tekstą. Šie atributai sudaro RTF dokumento antraštę.

Pateiktame RTF dokumento pavyzdyje (5 pav.) frazė "Čia paprastas tekstas." nėra grupės dalis, todėl traktuojamas kaip dokumento tekstas.

Pagal RTF specifikaciją galima kurti RTF konverterius (angl. *RTF reader*, *RTF writer*). Kai dokumentas yra išsaugomas kaip RTF formatu, konverteris (*RTF writer*) pakeičia tekstų doroklio žymes į RTF kalbą. Ir atvirkščiai, kai dokumentas yra atidaromas, konverteris (*RTF reader*) išverčia RTF kalbą į tekstų doroklio žymių kalbą.

```
{\rtf\ansi\deff0{\fonttbl{\f0\froman Tms Rmn;}{\f1\fdecor
Symbol;}{\f2\fswiss Helv;}}{\colortbl;\red0\green0\blue0;
\red0\green0\blue255;\red0\green255\blue255;\red0\green255\
blue0;\red255\green0\blue255;\red255\green0\blue0;\red255\
green255\blue0;\red255\green255\blue255;}{\stylesheet{\fs20
\snext0Normal;}}{\info{\author Audrys Kažukauskas}
{\creatim\yr1990\mo7\dy30\hr10\min48}{\version1}{\edmins0}
{\nofpages1}{\nofwords0}{\nofchars0}{\vern8351}}\widoctrl\ft
nbj \sectd\linex0\endnhere \pard\plain \fs20 Čia paprastas
tekstas.\par}
```

5 pav. RTF dokumento pavyzdys

1.5.2 PDF formatas

PDF (angl. *Portable Document Format*) – tai nuo platformos nepriklausantis binarinis failo formatas, sukurtas Adobe Systems. PDF formatas leidžia aprašyti dokumentus, kurie sudaryti iš bet kokios teksto, grafikos bei paveikslėlių kombinacijos.

PDF formatas yra trijų technologijų derinys [19]:

- ✓ apkarpyta PostScript technologija dokumentui maketuoti ir grafikai generuoti;
- ✓ šriftų įterpimo/pakeitimo technologija, leidžianti šriftams keliauti kartu su dokumentais. Ši technologija užtikrina dokumento skaitomumą, net jeigu skaitytojas ir nėra įsidiegęs į dokumente nurodytų šriftų;
- ✓ struktūrizuota išsaugojimo technologija, kuri talpina visus šiuos elementus į viena failą, panaudodama duomenų suspaudimo mechanizmą.

Dokumentai gali būti labai sudėtingi, naudojantys daug šriftų, spalvų bei grafikos. PDF dokumento puslapių skaičius nėra ribojamas.

PDF specifikacija PDF dokumentą skaitančiai programai draudžia jo turinį interpretuoti ir rodyti kitaip nei jis buvo originaliai sukurtas. PDF dokumento autorius gali būti tikras, kad jo sukurtas dokumentas tiek ekrane, tiek atspausdintas atrodys vienodai. Tai nepriklauso nei nuo kompiuterio ar spausdintuvo tipo, nei nuo jį sukūrusios ar skaitančios programos. Dažniausiai PDF dokumentai yra kuriami tik skaityti.(angl. *read-only*), t.y. autorius sukuria dokumentą ir jį išplatina be teisės kitiems asmenims jį redaguoti.

Keičiant PDF dokumento vaizdo mastelį, yra išlaikomas originalus dokumento maketavimas, o visi dokumento elementai didėja ar mažėja vienodai proporcingai.

1.5.3 DOC formatas

DOC yra binarinis failo formatas, kurį naudoja biuro programų paketui Microsoft Office priklausantis teksto doroklis Microsoft Word. Šis tekstų doroklis šiuo metu yra dominuojantis, tad DOC yra tapęs *de facto* dokumentų formato standartu. Konkuruojantys produktai (AbiWord, OpenOffice.org ir kt.) stengiasi užtikrinti suderinamumą su DOC formato dokumentais, nors tai realizuoti yra sunku. Suderinamumo tenka siekti analizuojant DOC formato dokumentų struktūrą (angl. *reverse engineering*), nes Microsoft korporacijos pateikta šio failo formato specifikacija yra nepilna ir nenuosekli. Programinių priemonių DOC formato binariniams failams skaityti yra nedaug. Viena iš jų yra atviro kodo Java biblioteka *Apache Jakarta POI*.

DOC formatas nėra pritaikytas duomenims apsikeisti, todėl pasikeitimas duomenimis šiuo formatu tarp įvairių įrenginių ir platformų nėra įmanomas. Stengiantis pašalinti šį trūkumą, naujausia Microsoft Word 2003 versija siūlo galimybę konvertuoti dokumentus iš DOC binarinio formato į XML formatą.

1.5.4 SGML formatas

SGML (angl. *Standard Generalized Markup Language*) yra metakalba, kuri leidžia apibrėžti dokumentų žymių kalbą [20]. SGML išsivystė iš GML (angl. *Generalized Markup Language*). SGML dokumentai yra tekstinio formato.

SGML kalba buvo kuriama specialiai vyriausybiniams ir kosmonautikos projektams, kuriuose reikėjo mechanizmo kompiuterių skaitomiems dokumentams apsikeisti.

```
<CITATA TIPAS="pavyzdys">
 SGML yra kažkas panašaus į <KURSYVAS>tai</KURSYVAS>
</CITATA>
```

6 pav. SGML dokumento pavyzdys

SGML yra lanksti ir galinga kalba, tačiau jos sudėtingumas buvo pagrindinė priežastis, kuri užkirto kelią platesniam jos paplitimui mažesnės apimties projektuose. Iš SGML kalbos išsivystė HTML ir XML kalbos.

1.5.5 HTML formatas

HTML (angl. *HyperText Markup Language*) yra žymių kalba, specialiai pritaikyta kurti interneto puslapius, t.y. informacijai, kuri pateikiama interneto Voratinklyje (angl. *World Wide Web*) [21]. Šiuo metu naujausia versija yra HTML 4.0.1.

HTML kalba yra apibrėžiama kaip konkretus SGML taikomasis atvejis. HTML dokumentai yra tekstinio formato.

HTML turi 4 žymių rūšis:

- ✓ *struktūrinės* žymės aprašo teksto tikslą (pvz., `<h1>Krepšinis</h1>` verčia skaitytoją žodį „Krepšinis“ traktuoti kaip pirmo lygio antraštę);
- ✓ *pateikimo* žymės aprašo vizualų teksto pateikimo būdą (pvz., `<b>paryškintas tekstas</b>` bus rodomas kaip „**paryškintas tekstas**“).
- ✓ *hipertekstinės* žymės, kurios sieja vienus dokumentus su kitais (pvz., `<a href="http://ifko.ktu.lt/">Kompiuterių katedra</a>` bus rodomas kaip tekstas Kompiuterių katedra, kuris yra hipernuoroda į Kompiuterių katedros internetinį puslapį adresu http://ifko.ktu.lt/);
- ✓ *funkciniai naudotojo sąsajos elementai* (angl. *widgets*), skirti objektams (pvz., mygtukams) kurti.

Interneto naršyklė skaitydama HTML dokumentą orientuojasi pagal HTML žymes, kurios instruktuoja, kaip turėtų būti rodomas dokumento turinys.

1.5.6 XML formatas

XML yra universali meta žymių (angl. *meta-markup*) kalba, sukurta *World Wide Web Consortium* (*W3C*) ir skirta aprašyti struktūrizuotiems duomenims [5]. Kuriant XML kalbą stengtasi išvengti pagrindinių jos protėvių HTML ir SGML trūkumų. XML kalbą galima traktuoti kaip supaprastintą SGML kalbos poaibį. Kaip ir SGML atveju, XML dokumentai taip pat yra tekstinio formato.

HTML pasižymi griežtai apibrėžta žymių (angl. *tags*) semantika. Laikui bėgant į HTML buvo įtraukiama vis daugiau funkcionalumo, žymių neišvengiamai daugėjo ir kalba pamažu tapo sudėtinga ir griozdiška.

SGML buvo naudojama nuo 8-ojo dešimtmečio. Jos žymės nėra semantiškai griežtai apibrėžtos. Tačiau SGML turėjo kitų trūkumų: ji buvo pernelyg sudėtinga, nebuvo paplitus, o SGML parserio (angl. *parser*) programavimas brangiai kainavo.

XML šių problemų neturi. Nors XML yra nėra labai paprasta kalba, tačiau yra plačiai paplitusi, yra sukurta daug XML apdorojimo įrankių (praktiškai kiekviena aukštesnio lygio programavimo kalba turi bent po keletą XML parserių). Be to, XML dokumento autorius pats gali kurti žymes ir jų semantiką pritaikyti konkrečiai programai.

```
<Klientas KlientoId="543">
        <Pavadinimas>Baitas</Pavadinimas>
        <Adresas>Savanorių pr. 78</Adresas>
       <Miestas>Kaunas</Miestas>
        <Šalis>LT</Šalis>
        <PaštoIndeksas>3000</PaštoIndeksas>
</Klientas>
```

7 pav. XML dokumento pavyzdys

XML pasirinkimo DVS dokumentų atvaizdavimo formatu motyvai bei privalumai yra tokie:

1. *XML yra universali kalba*. XML universalumas aprašyti bet kokio tipo duomenis leidžia kompiuterinėms sistemoms turėti bendrinį metodą tiems duomenims apdoroti bei jais apsikeisti.
2. *XML yra save aprašanti kalba*. Kiekvienas XML dokumentas savyje turi ir duomenų struktūros aprašą. Jis slypi elementų žymėse. Tokiu būdu programai yra paprasčiau naudoti dokumentą, sugeneruotą kitos, nežinomos, programos.
3. *XML yra plačiai paplitusi*. XML yra bene labiausiai paplitęs duomenų aprašymo būdas. Be to, XML įsisavinimo kreivė yra palyginti lėkšta, tad įsisavinimas trunka neilgai.
4. *Didelė XML apdorojimo įrankių pasiūla.* XML paplitimas lemia didelę brandžių ir našių XML apdorojimo įrankių pasiūlą
5. *XML yra portabili*. XML dokumentais operuojančios programos gali veikti bet kokiose platformose. XML yra tekstinio formato, ne binarinio, ir nėra susietas su kokio nors konkretaus tipo kompiuteriu. Be to, XML naudoja unikodą, tad dokumentai gali būti rašomi bet kokia kalba.

6. *XML dokumentai yra lengvai papildomi*. Programuotojai gali nesunkiai papildyti XML dokumentą nauja informacija ir nesutrikdyti programų, kurios sukurtos operuoti ankstesnio formato XML dokumentu.
7. XML tinka aprašyti hierarchinius duomenis. Dokumentai kaip tik ir turi hierarchinę struktūrą.
8. *Žmonės gali skaityti XML*. XML yra skirta mašinoms, tačiau ir žmonės gali ją skaityti. Programuotojai gali lengviau rasti ir ištaisyti klaidas XML dokumentuose nei, pavyzdžiui, binariniuose formatuose.

XML turi ir trūkumų. Palyginus su binariniais formatais XML trūkumų yra trys ir visi jie yra susiję būtent su XML universalumu [22]:

1. *Dydis*. Dokumentų dydis yra svarbus faktorius juos išsaugant ar siunčiant Internetu. Kuo dydis mažesnis, tuo geriau. XML žymių aprašai užima nemažą nuošimtį viso naudingojo XML dokumento teksto turinio, ypač jei dokumentas yra mažesnis. Pavyzdžiui, klientą aprašančiame XML dokumente (7 pav.) žymės užima virš 50% viso dokumento turinio.

   Binarinio formato dokumentas yra mažesnis nei XML, nes binariniuose dokumentuose žymių arba iš viso nėra, arba žymės nėra tekstinės. Taigi XML dokumentui išsaugoti reikia daugiau vietos nei binariniam.
2. *Skaitymo našumas*. XML dokumentas skaitymas yra lėtesnis nei binarinių. Žymėms nuskaityti reikia papildomų laiko sąnaudų.
3. *Sudėtingumas*. XML dokumentą yra sudėtingiau skaityti nei binarinį, nes reikia apdoroti žymes. Be to, parašyti parserį XML yra sunkiau nei binariniam formatui, tačiau XML parserį galima naudoti pakartotinai.

Kaip buvo minėta, šie aukščiau paminėti XML trūkumai yra kaina, kuri mokama už XML universalumą.

Reikia atsižvelgti į dar vieną XML bruožą: *XML naudojimas padidina programos apimtį*. XML naudojimas gali supaprastinti programą. Tačiau pridedant XML parserį prie nedidelės programos, jos apimtis gali padidėti net keletą kartų.

Tačiau XML privalumai atsveria visus jo trūkumus [22].

XML dokumentų dydis nėra problema, nes diskai informacijai saugoti nuolat didėja ir pinga, Interneto pralaidumas auga.

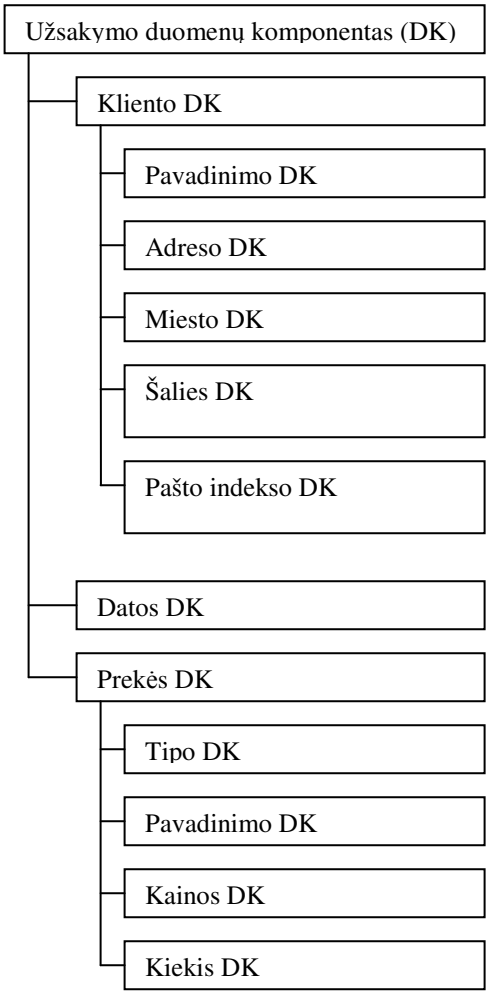
Kompiuteriai jau dabar yra pakankamai spartūs apdoroti XML dokumentus. XML dokumentų skaitymo sparta gali būti problema tik esant dideliam transakcijų skaičiui per laiko vienetą. Spartėjant kompiuteriams šis trūkumas ilgainiui išnyks savaime.

XML sudėtingumas jau dabar nebėra trūkumas, nes nėra būtinybės rašyti XML parserio patiems. Yra daug gatavų tiek komercinių, tiek nemokamų XML parserių.

XML kalba gerai tinka aprašyti tiek į duomenis orientuotus, tiek į dokumentus orientuotus dokumentus. XML dokumentus galima nesunkiai konvertuoti į kitokius žmogui dirbti patogesnius formatus (PDF, RTF)[6].

1.5.6.1 Į duomenis orientuotų dokumentų atvaizdavimas XML kalba

Į duomenis orientuotus dokumentus atvaizduoti XML kalba yra lengva, nes XML ir buvo sukurta tokiam tikslui, t. y. aprašyti struktūrizuotus duomenis. Kaip pavyzdį galima pateikti prekės užsakymą:

Paaiškinimai	XML dokumentas
	

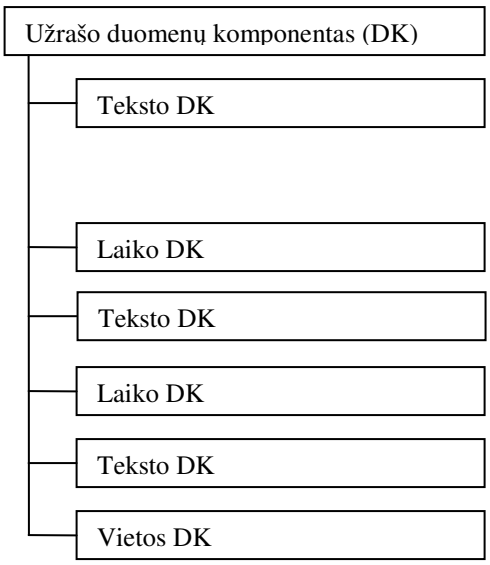

```
<Užsakymas SONumber="12345">
  <Klientas KlientoId="543">
     <Pavadinimas>Baitas</Pavadinimas>
     <Adresas>Savanorių pr. 78</Adresas>
     <Miestas>Kaunas</Miestas>
     <Šalis>LT</Šalis>
     <PaštoIndeksas>3000</PaštoIndeksas>
  </Klientas>
  <Data>20031215</Data>
  <Prekė PrekesId="1" PrekėsNr="123">
     <Tipas>Nešiojamas kompiuteris</Tipas>
     <Pavadinimas>IBM-DI32</Pavadinimas>
     <Kaina>3999.00</Kaina>
     <Kiekis>1</Kiekis>
  </Prekė>
</Užsakymas>
```

8 pav. Į duomenis orientuotas dokumentas, atvaizduotas XML

Kaip buvo minėta, į duomenis orientuotame dokumente duomenų komponentų eiliškumo tvarka paprastai nėra svarbi. Tačiau ši taisyklė galioja tik duomenų komponentams, kurie turi tą patį tėvą. Tokiems duomenų komponentams eiliškumas nėra svarbus. Pavyzdžiui, duomenų komponento *Klientas* vaikų (*Pavadinimas, Adresas, Miestas, Šalis* ir *PaštoIndeksas*) eiliškumas yra nesvarbus (8 pav.).

1.5.6.2 Į dokumentus orientuotų dokumentų atvaizdavimas XML kalba

XML kalba galima aprašyti ne tik griežtai struktūrizuotus, bet ir neturinčius aiškios struktūros duomenis. Į dokumentus orientuoto XML dokumento pavyzdys pateikiamas žemiau (9 pav.):

Paaiškinimai	XML dokumentas
	<Užrasas> <Tekstas> Svarbu! Keičiamas pirmadienį kviečiamo susirinkimo laikas, iš <Tekstas> <Laikas>8:00 val.</Laikas> <Tekstas> į </Tekstas> <Laikas>10:00 val.</Laikas> <Tekstas> Susirinkimo vieta: </Tekstas> <Vieta>18 kab.</Vieta> <Užrasas>

9 pav. Į dokumentus orientuotas dokumentas, atvaizduotas XML kalba

Priešingai į duomenis orientuotiems, į dokumentus orientuotų dokumentų duomenų komponentų eiliškumo tvarka yra labai svarbi (9 pav.). Jeigu šiame dokumente sukeitus vietomis *Laiko* duomenų komponentus, pasikeistų viso dokumento semantika.

1.5.7 Dokumentų atvaizdavimo formato parinkimas

Dokumentų formatus galima suskirstyti į dvi grupes:

- ✓ tekstiniai (RTF, HTML, SGML, XML),
- ✓ binariniai (DOC, PDF).

Binarinis DOC formatas nėra pritaikytas duomenims apsikeisti. Be to, detali DOC formato specifikacija nėra viešai prieinama, todėl sunku kurti programines priemones šio formato dokumentams skaityti ir rašyti. Dėl šių priežasčių DOC formatas nėra tinkamas naudoti kaip DVS dokumentų atvaizdavimo formatas.

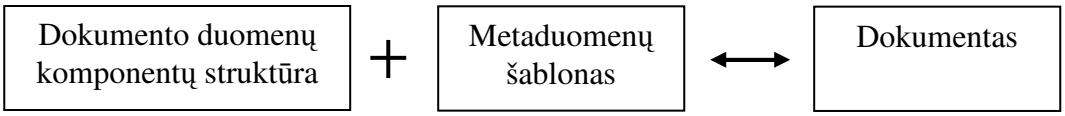
Binarinis PDF formatas remiasi PostScript technologija ir yra daugiau skirtas skaityti nei rašyti. Operuoti PDF dokumento turiniu būtų sudėtinga ir dėl naudojamo suspaudimo mechanizmo. Norint paimti duomenis iš PDF dokumento, reikėtų naudoti kompiuterinius resursus dokumentui išarchyvuoti, o dėl šios priežasties atsirastų vėlinimas. Be to, PDF formatas nenumato jokių galimybių kurti naujas žymes ir joms suteikti tam tikrą semantiką. Dėl šių priežasčių PDF formatas nėra tinkamas naudoti kaip DVS dokumentų atvaizdavimo formatas.

Tekstinių RTF ir HTML formatų žymių semantika taip pat yra griežtai apibrėžta, todėl nėra galimybės kurti naujas žymes. Dėl šių priežasčių RTF ir HTML formatas nėra tinkamas naudoti kaip DVS dokumentų atvaizdavimo formatas.

RTF ir HTML tipo formatai turi ir kitų trūkumų. RTF ir HTML formato dokumentuose duomenų komponentai visada yra neatskiriamai susieti su juos atitinkančiais metaduomenų komponentais (pvz., šrifto tipu ir dydžiu). Nėra galimybės duomenų komponentą atskirti nuo metaduomenų komponento. Paprastai organizacijose dokumentai yra skirstomi į kategorijas, pvz., posėdžių protokolai, nutarimai ir kt. Atskira kategorija gali turėti jai būdingą šabloną, kuriame nurodyta, kokie duomenys ir koks duomenų stilius turi būti visuose tos kategorijos dokumentuose. Jei organizacija kaupia dokumentus RTF ar HTML formatu, tai pasikeitus šablonui, tektų kiekvieną atskirą dokumentą redaguoti atskirai rankiniu būdu.

Kad išvengti tokių problemų, galima dokumentą suskirstyti į dvi struktūras: duomenų komponentų struktūrą ir metaduomenų komponentų struktūrą - metaduomenų šabloną. Galima išvesti paralelę su šviečiančia lempute ir keletu skirtingų spalvų filtrų. Uždengus lemputę žaliu filtru, gaunama žalia šviesa. Keičiant filtrų spalvas išgaunama vis kitokios spalvos šviesa, nors šviesos šaltinis lieka tas pats – lemputė. Šioje alegorijoje lemputė yra duomenų komponentų

struktūra, o šviesos filtrai - metaduomenų šablonai. Dokumento duomenų komponentų struktūra ir metaduomenų šablonas yra išsaugomi atskirai, o išrinkimo proceso metu dokumentas yra rekonstruojamas suliejant duomenų komponentų struktūrą su parinktu metaduomenų šablonu.



10 pav. Dokumento išskaidymas į duomenų komponentus ir jų atributus

Dokumento išskaidyti į duomenų komponentų struktūrą ir metaduomenų šabloną galima nesunkiai, jei dokumentas yra aprašytas SGML ar XML kalba [6].

Skirtingai nei RTF ar HTML, SGML ir XML formatai suteikia galimybę kurti atskiras dokumentų kalbas ir jų žymėms suteikti norimą semantiką. Kaip buvo minėta, XML kalbą galima traktuoti kaip supaprastintą SGML kalbos poaibį. XML kalbą kompiuteriams paprasčiau apdoroti nei SGML kalbą, be to, XML šiuo metu yra jau visiškai pakeitęs SGML kalbą kaip priemonę struktūrizuotiems duomenims aprašyti. Dėl šių priežasčių SGML, kaip potencialus DVS dokumentų atvaizdavimo formatas, toliau nėra nagrinėjamas.

Išanalizavus egzistuojančių dokumentų formatų privalumus ir trūkumus, šiame darbe daroma išvada, XML formatas yra tinkamiausias dokumentų formatas DVS. Toliau yra analizuojamos alternatyvios XML dokumentų saugyklos.

1.6 Dokumentų saugyklos

Yra dvi duomenų saugojimo alternatyvos: failinė sistema arba duomenų bazių valdymo sistema (DBVS). Failinė sistema puikiai tinka saugoti duomenims dokumentų pavidalu, jei dokumentų yra nedaug ir nėra poreikio vykdyti tekstinės paieškos keliuose dokumentuose iš karto. Tačiau esant didesniam dokumentų skaičiui ar plačiam naudotojų ratui neabejotinai patogiau kaip duomenų saugyklą naudoti DBVS.

DBVS suteikia DVS ne tik efektyvią dokumentų saugyklą, bet ir daugiau galimybių bei papildomų privalumų, tokių kaip:

- ✓ integralumas,
- ✓ saugumas,
- ✓ transakcijos,
- ✓ priėjimas prie dokumentų daugeliui naudotojų vienu metu,
- ✓ indeksai (didėja paieškos greitis),
- ✓ trigeriai
- ✓ paieška ne tik atskirame dokumente, bet ir dokumentų grupėje.

Palyginus failinės sistemos ir DBVS teikiamas galimybes, daroma išvada, kad duomenų bazių valdymo sistemos (DBVS) kaip dokumentų saugyklos yra pranašesnės prieš failinę sistemą.

Šiuo metu duomenų bazių valdymo sistemų tipų yra gana daug. Visos jos buvo kuriamos siekiant panašių tikslų, tačiau akcentuojant skirtingas savybes, todėl kiekvienas DBVS tipas turi savitų privalumų ir trūkumų. Šiame skyriuje bus pateikiama DBVS tipų, kurie yra svarstytini XML dokumentams saugoti, apžvalga.

1.6.1 Objektinės duomenų bazių valdymo sistemos

Objektinės DBVS saugo objektus bei jų atributus ir elgseną. Pastaruoju metu vyksta ginčai, ar šio tipo DBVS iš viso dar gyvuoja. Objektinės DBVS šiame darbe nėra analizuojamos, nes:

- ✓ objektai paremti enkapsuliacijos principu, o duomenims enkapsuliacija nėra taikoma;
- ✓ objektai pasižymi elgsena, o duomenys jos neturi.
- ✓ Objektinės DBVS yra retos ir nepaplitusios.

1.6.2 Hierarchinės duomenų bazių valdymo sistemos

Tai pats seniausias duomenų bazių valdymo sistemų tipas, sukurtas 6-jame dešimtmetyje. Hierarchinės DBVS duomenis saugo medžio struktūra.

Hierarchinių DBVS privalumai:

- ✓ Tinka realizuoti *1* x *1* (angl. one-*to-one*) ir *1* x *N* (angl. *one-to-many*) ryšius tarp duomenų, kai duomenys yra vaizduojami medžio struktūra.

Hierarchinių DBVS trūkumai:

- ✓ Hierarchinių ryšių nustatymas reikalauja papildomų manipuliacijų. Iš anksto nenumatytos užklausos gali pareikalauti modelio perorganizavimo [13];
- ✓ Netinka realizuoti *N* x *M* (angl. *many-to-many*) ryšius tarp duomenų, kai $N > 1, M > 1$;
- ✓ Neefektyviai naudojama atmintis, kadangi duomenys kartojasi. Iš dalies atminties galima sutaupyti panaudojant virtualius objektus ir rodykles, kurie atminties neeikvoja, nes jų turinys: adresas, kur saugomi duomenys [13];
- ✓ Duomenų struktūra gali būti tik medžio formos;
- ✓ Nėra paplitusios.

Hierarchinės DBVS tinka saugoti į dokumentus orientuotus dokumentus, tačiau dokumentų valdymo sistemos operuoja ir į duomenis orientuotais dokumentais. Efektyviai išsaugoti pastarojo tipo dokumentų dėl aukščiau paminėtų architektūrinių trūkumų hierarchinė DBVS negali. Dėl šių priežasčių objektinės DBVS nėra plačiau analizuojamos.

1.6.3 XML duomenų bazių valdymo sistemos

XML DBVS yra palyginti naujas DBVS tipas [14, 22]. Jos kuriamos vienam tikslui: XML dokumentui išsaugojimui. Fundamentalus šio tipo DBVS saugojimo vienetas yra XML dokumentas. Dažnos XML duomenų bazės kaip saugyklas naudoja kitų tipų DBVS: sąryšines, hierarchines ir kt. [14, 22].

XML DBVS dar nėra brandi technologija, paprastai jai trūksta elementų lygio blokavimo mechanizmo (angl. *locking*), patikimumo, daug kompiuterinių resursų reikalauja duomenų indeksavimas. Dėl šių priežasčių XML DBVS šiame darbe nėra plačiau analizuojama.

1.6.4 Sąryšinės duomenų bazių valdymo sistemos

Sąryšinės duomenų bazių valdymo sistemos SDBVS buvo sukurtos 8-jame dešimtmetyje, o šiuo metu užima didžiausią DBVS rinkos dalį. Tai laiko išbandyta, brandi technologija tiek stabilumo, tiek galimybių atžvilgiu.

SDBVS technologija pagrįsta sąryšiniu (angl. *relational*) modeliu. Puikiai tinka saugoti griežtai struktūrizuotus duomenis (į duomenis orientuotus dokumentus), lengvai tvarkosi su visais ryšių tipais: vienas-vienas (angl. *one-to-one*), vienas-daug (angl. *one-to-many*), daug–daug (angl. *many-to-many*) ryšiais.

1.6.5 Dokumentų saugyklos parinkimas

Atlikus dokumentų saugyklų analizę, paaiškėjo, kad XML duomenų bazių valdymo sistemos dar nėra pakankamai brandžios, kad galėtų sudaryti alternatyvą sąryšinėms duomenų bazių valdymo sistemoms. Šiame darbe daroma išvada, kad XML dokumentams išsaugoti geriausiai tinka sąryšinės duomenų bazių valdymo sistemos.

1.7 Dokumentų išsaugojimo ir išrinkimo metodai

Kadangi į duomenis orientuotų XML dokumentų struktūra yra griežtai apibrėžta, juos labai tinka saugoti egzistuojančiose sąryšinėse duomenų bazių valdymo sistemose (SDBVS).

Prieš dokumentą išsaugant SDBVS, tereikia jo struktūrą pakeisti į tinkamą SDBVS struktūrą. Analogiškai, dokumentą išrenkant, tereikia dokumentą iš SDBVS struktūros pakeisti į jo pirminę struktūrą.

Į dokumentus orientuotus XML dokumentus siūloma skaidyti į atskirus XML elementus [17]. XML elementai su visais tarpusavio ryšiais yra išsaugomi sąryšinėje duomenų bazių valdymo sistemoje. Kaip buvo minėta, Į dokumentus orientuoti dokumentai gali būti atvaizduojami orientuotu grafu, kurio viršūnės atitinka dokumento duomenų komponentus. Grafo viršūnės atitinka XML dokumentų elementus.

XML dokumentų išsaugojimo SDBVS metodus galima suskirstyti į dvi grupes. Vieni metodai saugo grafo viršūnes (angl. *mapping values*), kiti – grafo briaunas (angl. *mapping edges*)[17].

Toliau šiame darbe tiriamas antrai grupei priklausantis standartinis briaunų (angl. *Edge*) metodas.

1.7.1 Standartinis briaunų metodas

Naudojant standartinį briaunų metodą, XML dokumentas atvaizduojamas orientuotu grafu, kurio viršūnės atitinka XML elementus. Kiekvienai grafo viršūnei priskiriamas unikalus identifikacinis numeris. Ryšys tarp tėvinės ir dukterinės viršūnės yra vaizduojamas briauna, kuri vadinama dukterinės viršūnės vardu. Visos dokumento briaunos yra išsaugomos vienoje sąryšinės duomenų bazės lentelėje.

Atlikti tyrimai rodo [17], kad standartinis briaunų metodas yra neefektyvus, kai atliekamos stambios užklausos. Pavyzdžiui, pilna XML dokumento rekonstrukcija reikalauja atlikti daug apjungimo operacijų (angl. *joins*).

Tarkime, duotas tekstinis dokumentas (11 pav.):

```
Už puikius darbo rezultatus Jonas Jonaitis
apdovanojamas medaliu.

Direktorius
Petras Petraitis
```

11 pav. Į dokumentus orientuotas dokumentas

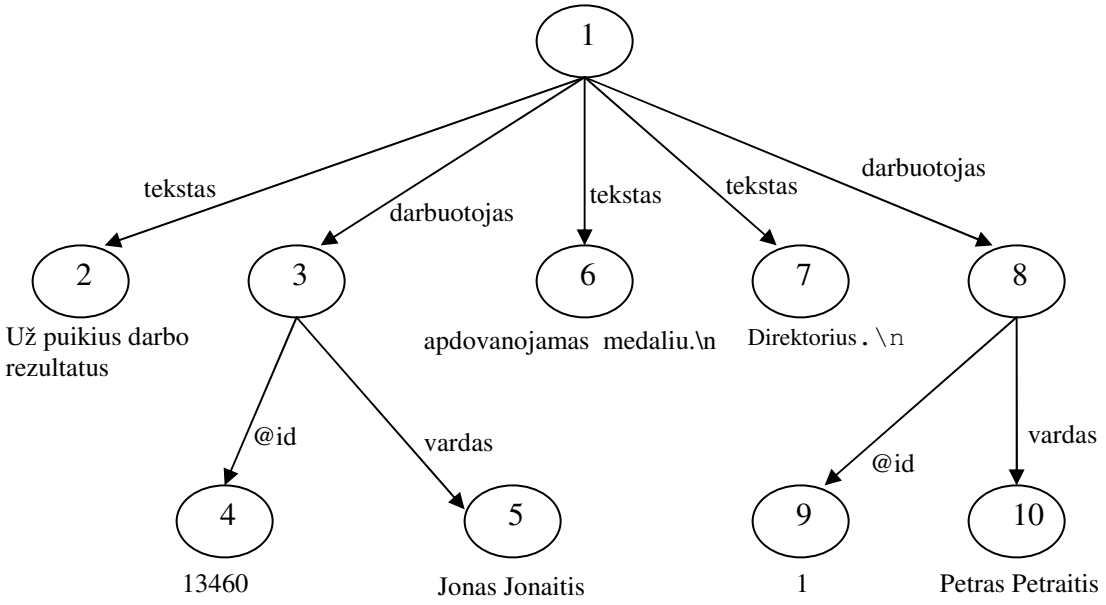
DVS tokį dokumentą atvaizduoja XML formatu: išskaido tekstą į atskirus duomenų komponentus ir juos transformuoja į XML kalbos elementus. Tarkime, gaunamas toks į dokumentus orientuotas XML dokumentas pavadinimu *Pagyrimas210.xml* (12 pav.). Dokumento struktūra nėra griežtai apibrėžta, todėl jam negalima taikyti nei *DTD* [15] nei *XML Schema* [16] validacijos mechanizmų:

```
<pagyrimas>
  <tekstas> Už puikius darbo rezultatus</tekstas>
  <darbuotojas id=”13460”> Jonas Jonaitis</darbuotojas>
  <tekstas> apdovanojamas medaliu\n</tekstas>
  <tekstas> Direktorius\n\n</tekstas>
  <darbuotojas id=”1”> Petras Petraitis</darbuotojas>
</pagyrimas>
```

12 pav. XML dokumentas *Pagyrimas210.xml*

Tokį dokumentą galima pavaizduoti orientuotu grafu (13 pav.).

Standartinis briaunų metodas įgalina bet kokį XML dokumentą (tiek į dokumentus orientuotą, tiek į duomenis orientuotą) išsaugoti SDBVS. Visos XML dokumento grafo briaunos yra išsaugomos vienoje lentelėje Įrašai.



13 pav. XML dokumento *Pagyrimas210.xml* grafas

Pateikiamas anksčiau nagrinėto XML dokumento (12 pav.) atvaizdavimas į SDBVS standartiniu briaunų metodu (1 lentelė, 2 lentelė).

1 lentelė. Duomenų bazės lentelė *Įrašai*

Įrašai						
Id	**DokumentoID**	**TėvasID**	**Pavadinimas**	**Reikšmė**	**EilėsNr**	**Tipas**
1	1	0	pagyrimas	null	1	0
2	1	1	tekstas	Už puikius darbo rezultatus	1	1
3	1	1	darbuotojas	null	2	0
4	1	3	@id	13460	1	1
5	1	3	vardas	Jonas Jonaitis	2	1
6	1	1	tekstas	apdovanojamas medaliu.\n	3	1
7	1	1	tekstas	Direktorius.\n	4	1
8	1	1	darbuotojas	null	5	0
9	1	8	@id	1	1	1
10	1	8	vardas	Petras Petraitis	2	1

2 lentelė. Duomenų bazės lentelė *Dokumentai*

Dokumentai	
Id	**Pavadinimas**
1	Pagyrimas210.xml

Lentelėje *Dokumentai* laikomi atskirų XML dokumentų unikalus identifikaciniai numeriai *Id* ir pavadinimai *Pavadinimas*.

Lentelėje *Įrašai* laikomos visos XML dokumento(-ų) viršūnės : rodyklės, elementai ir jų atributai. Stulpelių reikšmės:

1. *Id* yra unikalus kiekvienos viršūnės identifikacinis numeris;
2. *DokumentoID* yra atitinkamo dokumento identifikacinis numeris;
3. *TėvasId* yra viršūnės tėvinio elemento identifikacinis numeris (*Dokumentai.Id = Įrašai.DokumentoId*);
4. *Pavadinimas* – viršūnės pavadinimas. Simbolis @ žymi atributą;
5. *Reikšmė* – viršūnės reikšmė (null, jei nėra);
6. *EilėsNr* – žymi viršūnės poziciją jai lygiagrečių viršūnių, turinčių tą pačią tėvinę viršūnę, atžvilgiu.
7. *Tipas* – viršūnės tipą nusakanti vėliavėlė. Reikšmė lygi 0, jei viršūnė yra rodyklė. Reikšmė lygi 1, jei viršūnė yra elementas arba atributas.

2. Teorinis dokumentų išsaugojimo ir išrinkimo procesų modelis

Šiame darbe siūloma modifikuoto briaunų metodo koncepcija, kuri turėtų išvengti standartinio metodo trūkumų. Pateikiama modifikuotojo briaunų metodo koncepcija ir konkretaus XML dokumento (12 pav.) išsaugojimo SDBVS pagal šį metodą pavyzdys.

Siūloma optimizuoti standartinį briaunų metodą atliekant tokius pakeitimus:

1. Sukuriama nauja lentelė *XKeliai* (3 lentelė), kurioje laikomi keliai nuo šakninės viršūnės iki atskirų atitinkamo pavadinimo viršūnių ir gyliai, kurie žymi atstumą nuo šakninės viršūnės. Kelias sudarytas iš jį sudarančių briaunų pavadinimų. Kiekvienas lentelės *XKeliai* įrašas yra susiejamas ryšiais *1xN* su lentelės *Įrašai* įrašais, kur $N \geq 1$.
2. Prie lentelės Įrašai pridedami papildomi stulpeliai (4 lentelė)
    a. *XKeliasId* – raktas į lentelę *XKeliai*.
    b. *Kelias* – unikalus kelias nuo šakninės viršūnės iki atitinkamos viršūnės.

3 lentelė. Duomenų bazės lentelė *XKeliai*

XKeliai		
Id	**Kelias**	**Gylis**
1	/pagyrimas	1
2	/pagyrimas/tekstas	2
3	/pagyrimas/darbuotojas	2
4	/pagyrimas/darbuotojas/@id	3
5	/pagyrimas/darbuotojas/vardas	3

Briaunų metodo papildymas lentele *XKeliai* leidžia realizuoti paiešką pagal *XPath* standartą ir suteikia galimybę vykdyti paiešką tarp nurodytų lygių ir pavadinimų viršūnių, pvz. */pagyrimas/darbuotojas/vardas[Jonas Jonaitis]*, t. y. surasti pagyrimuose visus darbuotojus, kurių vardas yra *Jonas Jonaitis* (ieškoma bus tik trečiajame lygyje nuo šakninės viršūnės). Šio patobulinimo tikslas yra paspartinti paieškos greitį XML dokumente, sumažinant perrenkamų įrašų skaičių.

4 lentelė. Duomenų bazės lentelė *Įrašai*

Įrašai							
Id	**Dokumento ID**	**TėvasID**	**Pavadinimas**	**Reikšmė**	**Tipas**	**XKeliasId**	**Kelias**
1	1	0	pagyrimas	null	0	1	/1
2	1	1	tekstas	Už puikius darbo rezultatus	1	2	/1/2
3	1	1	darbuotojas	null	0	3	/1/3
4	1	3	@id	13460	1	4	/1/3/4
5	1	3	vardas	Jonas Jonaitis	1	5	/1/3/5
6	1	1	tekstas	Apdovanojamas medaliu.\n	1	2	/1/4
7	1	1	tekstas	Direktorius.\n	1	2	/1/5
8	1	1	darbuotojas	null	0	3	/1/8
9	1	8	@id	1	1	4	/1/8/9
10	1	8	vardas	Petras Petraitis	1	5	/1/8/10

Prie lentelės *Įrašai* pridėtas stulpelis *Kelias* – unikalus kelias nuo šakninės viršūnės iki atitinkamos viršūnės, sudarytas iš kelyje esančių viršūnių identifikacinių numerių. Tai turėtų paspartinti viršūnių įterpimo ir šalinimo, o ypatingai atskiros XML dokumento dalies (viršūnės su visais jos vaikais) ar viso dokumento rekonstrukcijos operacijas.

3. Eksperimentinis modelis standartiniam ir modifikuotajam briaunų metodams palyginti

Šiame skyriuje aprašomas eksperimentinis modelis, kuris naudojamas praktiškai palyginti standartinio ir modifikuotojo briaunų metodus.

Eksperimente naudojamas vienas XML dokumentas, kurio parametrai pateikti žemiau (5 lentelė):

5 lentelė. Eksperimente naudojamo XML dokumento parametrai

n	~200000	Elementų skaičius
a_{max}	5	Maksimalus elemento atributų skaičius
d	15	Skirtingų atributų pavadinimų skaičius
l_e	50	Maksimalus elemento ar jo atributo pavadinimo ilgis
l_r	255	Maksimalus ilgis elemento arba atributo reikšmės ilgis
t	8	Maksimalus elementų lygių skaičius

Sukurtos dvi atskiros duomenų bazės DB_s ir DB_m. DB_s struktūra suformuota pagal standartinį, o DB_m - pagal modifikuotąjį briaunų metodą. DB_s duomenų bazės lentelės *Įrašai* stulpeliams *TėvasId* ir *Pavadinimas* suformuoti indeksai kaip nurodyta [22]. DB_m duomenų bazėje papildomai suformuoti indeksai stulpeliams *XKeliai.Kelias* ir *Įrašai.Kelias*. Abiejose duomenų bazėse išsaugotas tas pats aukščiau aprašytas XML dokumentas.

Eksperimento metu DB_s ir DB_m vykdomos vienodos užklausos, kurios manipuliuoja XML dokumento elementais ir objektais. XML objektu vadinama XML dokumento dalis (arba visas dokumentas), kuri apima vieną pradinį elementą su visais jo dukteriniais elementais. Toks pradinis elementas vadinamas objekto tėvine viršūne.

Eksperimente naudotos užklausos apibrėžtos 6 lentelėje:

6 lentelė. Užklausų apibrėžimai

UŽKLAUSA-1	rekonstruoti XML objektą pagal nurodytos objekto tėvinės viršūnės identifikacinį numerį
UŽKLAUSA-2	išrinkti XML elementus, kurių reikšmės yra lygios tam tikrai reikšmei a_1 arba a_2
UŽKLAUSA-3	pašalinti XML elementus, kurių identifikacinis numeris yra lygus tam tikrai reikšmei a_1. Elementai šalinami su visais jų dukteriniais elementais

Bandymui naudotas kompiuteris (procesorius AMD Duron 900MHZ, 256MB operatyvinė atmintis, 7200 aps./min. 60GB standusis diskas) su operacine sistema MS Windows 2000 Professional ir sąryšine duomenų bazių valdymo sistema MS SQL Server 2000.

Metodų įvertinimo kriterijus – trukmė, per kurį įvykdomos užklausos. Kiekviena užklausa duomenų bazėse DB_s ir DB_m vykdoma k kartų, kai $k = 10$. Kiekvieną kartą nustatoma užklausos trukmė t_i:

$$t_i = t_g - t_p ,$$

kur t_p – užklausos vykdymo pradžios laikas,

t_g – užklausos vykdymo pabaigos laikas,

t_i – i-tojo karto užklausos vykdymo trukmė, $i = 1..k$.

Po to apskaičiuojamas kiekvieno tipo užklausos vidutinė vykdymo trukmė t_{vid}:

$$t_{vid} = \frac{\sum_{1}^{k} t_i}{k} ,$$

kur t_i – i-tojo karto užklausos vykdymo trukmė,

k – užklausos vykdymo kartų skaičius, $k = 10$.

3.1 Eksperimento rezultatai

Pateikiamos eksperimento metu užfiksuotos vidutinės užklausų vykdymo trukmės pagal standartinį ir modifikuotąjį briaunų metodą duomenų bazėse DB_s ir DB_m (7 lentelė, 14 pav.):

7 lentelė. Vidutinė užklausų vykdymo trukmė

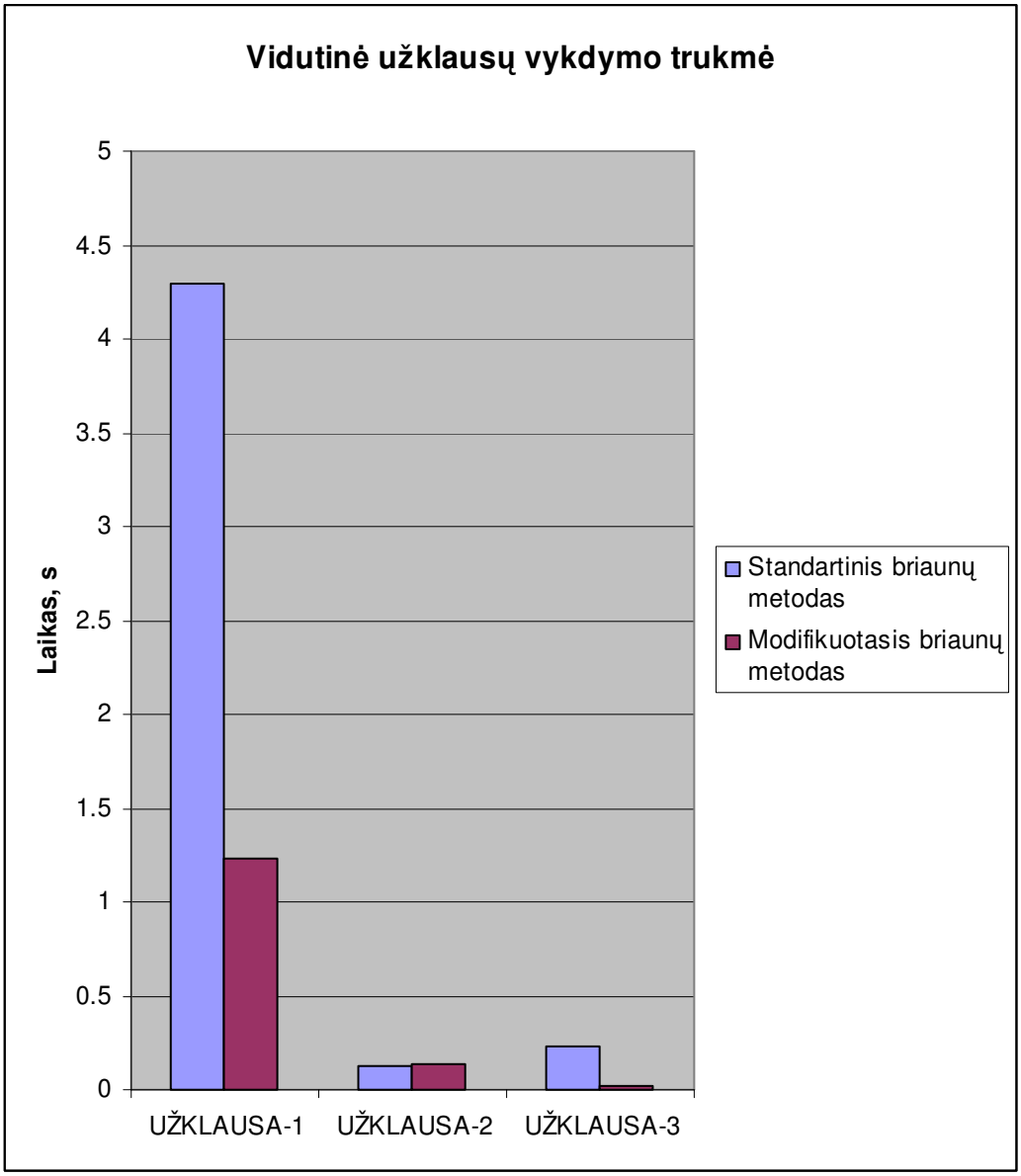
	t_{vid} , s	
	Standartinis briaunų metodas	**Modifikuotasis briaunų metodas**
UŽKLAUSA-1	4.294	1.233
UŽKLAUSA-2	0.129	0.133
UŽKLAUSA-3	0.231	0.019

Lyginant gautus duomenis galima spręsti apie standartinio ir modifikuotojo briaunų metodų efektyvumą. Modifikuotasis briaunų metodas ženkliai lenkia standartinį pagal XML objektų rekonstrukcijos trukmę (7 lentelė, 14 pav., UŽKLAUSA-1). Kai žinomas tėvinės XML objekto viršūnės identifikacinis numeris, rekonstruoti objektą pagal modifikuotą briaunų metodą pakanka vienos užklausos, nes XML objektą sudarantys elementai išrenkami pagal stulpelį *Įrašai.Kelias*. Standartinis metodas šiuo atveju su lentele *Įrašai* atlieka daug apjungimo operacijų (angl. *joins*), todėl užklausa trunka ilgiau.

Antrojo tipo užklausą (7 lentelė, 14 pav., UŽKLAUSA-2) abu metodai įvykdo per vienodą trukmę. Tai galima paaiškinti identiška DB_s ir DB_m duomenų bazių struktūra šios tipo užklausos atžvilgiu. Kadangi reikia išrinkti XML elementus, kurių reikšmės yra lygios tam tikrai reikšmei a_1 arba a_1, paieška vykdoma lentelių *Įrašai* stulpeliuose *Reikšmės*, kurie abejose duomenų bazėse yra vienodi.

Atliekant elementų šalinimo operacijas vėl ženkliai spartesnis yra modifikuotasis metodas (7 lentelė, 14 pav., UŽKLAUSA-3). Pastaruoju metodu visą elementų medį galima pašalinti įvykdžius vienintelę paprastą užklausą pagal stulpelį *Įrašai.Kelias*, tuo tarpu standartinis metodas vėlgi naudoja daug apjungimo operacijų.

Eksperimento rezultatai patvirtina prielaidą, kad modifikuotasis briaunų metodas yra spartesnis už standartinį briaunų metodą atliekant XML dokumento rekonstrukcijos ir elementų šalinimo operacijas.



14 pav. Vidutinė užklausų vykdymo trukmė

Išvados

- ✓ Tyrimo metu išspręsti darbui kelti uždaviniai:
    1. Atliktos egzistuojančių dokumentų atvaizdavimo formatų analizės metu nustatyta, kad XML yra efektyviausias DVS dokumentų atvaizdavimo formatas.
    2. Atlikta alternatyvių dokumentų saugyklų savybių analizė. Kaip efektyviausia DVS dokumentų saugykla parinkta sąryšinė duomenų bazių valdymo sistema.
    3. Pasiūlyta nauja briaunų metodo modifikacija XML dokumentams išsaugoti sąryšinėje duomenų bazių valdymo sistemoje. Atliktas eksperimentas įrodo modifikuoto briaunų metodo pranašumą prieš standartinį briaunų metodą, atliekant XML dokumentų rekonstrukcijos ir jų elementų šalinimo operacijas.
- ✓ Pagrindinis darbo tikslas pasiektas – pasiūlytas metodas (modifikuotasis briaunų metodas) tekstiniams dokumentams išsaugoti ir išrinkti XML formatu.
- ✓ Eksperimentiškai įrodyta, kad tikslinga kurti DVS, paremtą XML dokumentų saugojimu sąryšinėje duomenų bazių valdymo sistemoje.

Literatūra

1. Promin Computer Technologies [interaktyvus]. What is document management?. – [žiūrėta 2004-02-15]. Prieiga per internetą: <http://www.promin.com.tr/edms/whatisdms.htm>
2. Tcl DeveloperXchange [interaktyvus]. XML Introduction, 2001 m. balandis. – [žiūrėta 2004-05-09]. Prieiga per internetą: <http://www.tcl.tk/advocacy/xmlintro.html>
3. Apache XML Project [interaktyvus]. Understanding Apache Cocoon. – [žiūrėta 2004-04-09]. Prieiga per internetą: <http://cocoon.apache.org/2.0/userdocs/concepts/index.html>
4. ISYS search software [interaktyvus]. - [žiūrėta 2003-10-23]. Prieiga per internetą: <http://www.isysusa.com/products/whitepapers/ISYSsdkwhitepaper.pdf>
5. W3C [interaktyvus]. Extensible Markup Language (XML). Prieiga per internetą: <http://www.w3.org/XML/>
6. Apache XML Project [interaktyvus]. Overview of Apache Cocoon. – [žiūrėta 2004-04-09]. Prieiga per internetą: <http://cocoon.apache.org/2.0/overview.html>
7. Wikipedia [interaktyvus]. Tree data structure. – [žiūrėta 2003-05-03]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Tree_data_structure>
8. [interaktyvus]. What is document management?. – [žiūrėta 2003-10-23]. Prieiga per internetą: <www.documation.co.uk/solutions/What%20is%20Document%20Management.pdf>
9. Lietuvos jaunųjų matematikų mokykla [interaktyvus]. Grafai. – [žiūrėta 2004-05-11]. Prieiga per internetą: <http://www.maf.vu.lt/ljmm/kursa/sept/septuzd.html>
10. Juozapavičius A. Duomenų struktūros ir algoritmai. V.: Vilniaus universiteto leidykla, 1997. 15 psl.
11. Wikipedia [interaktyvus]. Graph (mathematics). – [žiūrėta 2004-05-03]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Directed_graph>
12. Wolfram Research [interaktyvus]. Graph thickness. – [žiūrėta 2004-05-03]. Prieiga per internetą: <http://mathworld.wolfram.com/GraphThickness.html>

13. [interaktyvus]. Duomenų bazių technologijų vystymasis. – [žiūrėta 2004-03-24]. Prieiga per internetą: <http://www.ik.ku.lt/lessons/konspekt/db/turinys/db_technologijos.htm>

14. [interaktyvus]. XML Database Products. – [žiūrėta 2003-11-07]. Prieiga per internetą: <http://www.rpbourret.com/xml/XMLDatabaseProds.htm>

15. W3Scools [interaktyvus]. DTD Tutorial. – [žiūrėta 2004-05-18]. Prieiga per internetą: <http://www.w3schools.com/dtd/default.asp>

16. W3C [interaktyvus]. XML Schema. – [žiūrėta 2004-05-18]. Prieiga per internetą: <http://www.w3.org/XML/Schema>

17. D. Florescu, D. Kossman. Storing and Querying XML Data using an RDBMS, Data Engineering Bulletin, 1999.

18. MSDN [interaktyvus]. Rich Text Format (RTF) Specification, version 1.6 – žiūrėta [2004-05-02]. Prieiga per internetą: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec_1.asp>

19. Wikipedia [interaktyvus]. Portale Document Format. – [žiūrėta 2004-05-02]. Prieiga per internetą: <http://en.wikipedia.org/wiki/Portable_Document_Format>

20. Wikipedia [interaktyvus]. Standard Generalized Markup Language. – [žiūrėta 2004-05-02]. Prieiga per internetą: <http://en.wikipedia.org/wiki/SGML>

21. Wikipedia [interaktyvus]. HyperText Markup Language. Prieiga per internetą: < http://en.wikipedia.org/wiki/Html>

22. rpbourret.com [interaktyvus]. XML and Databases. 2003 m. lapkritis – [žiūrėta 2004-01-12] <http://www.rpbourret.com/xml/XMLAndDatabases.htm>

Summary

Document management systems allow organizations to have greater control over the lifecycle of documents from creation through review, storage, retrieval and dissemination all the way to their destruction. Document management provides greater efficiencies in the ability to classify and reuse information.

This document deals with issues of storage and retrieval processes of text documents in document management systems. Main focus is made on choosing an effective document format, means and methods for storing and retrieving documents.

The paper suggests using *XML* as the base document format and relational database management system as backend storage of the document management system. A new modification of the standard *Edge* method for storing and retrieving *XML* documents from relational database management systems is introduced and the results of its performance experiment are presented. The experiment proves the performance superiority of the modified *Edge* method over its standard analogue.

Santrumpų ir terminų žodynas

1. **LOB –** (angl. *line-of-business*) – tai aibė kritinių kompiuterinių programų, kurios yra gyvybiškai svarbios organizacijos veiklai. Tai programos, susijusios su buhalterija, tiekimo grandinės vadyba (angl. *supply chain management – SCM*), resursų planavimu.
2. **XML** – angl. *Extensible Markup Language*, universali metakalba aprašyti struktūrizuotiems duomenims, kurie yra dažniausiai naudojama informacijai tarp programų perduoti ar apsikeisti.
3. **RTF** – angl. *Rich Text Format*, dokumentų formatavimo standartas, apibrėžtas Microsoft korporacijos.
4. **HTML** – angl. *HyperText Markup Language*, žymių kalba, specialiai pritaikyta kurti interneto puslapius, t.y. informacijai, kuri pateikiama interneto Voratinklyje (angl. *World Wide Web*). Šiuo metu naujausia versija yra HTML 4.0.1.
5. **SGML** – angl. *Standard Generalized Markup Language*, metakalba struktūrizuotiems duomenims aprašyti. Iš SGML išsivystė HTML ir XML kalbos.
6. **PDF** – angl. *Portable Document Format*, nuo platformos nepriklausantis binarinis failo formatas yra sukurtas Adobe Systems. PDF formatas leidžia aprašyti dokumentus, kurie sudaryti iš bet kokios teksto, grafikos bei paveikslėlių kombinacijos.